Journal of Computational and Applied Mathematics 45 (1993) 91–102
North-Holland

CAM 1279

On the numerical solution of a class of singular two-point boundary value problems

J.R. Cash and H.H.M. Silva *
Department of Mathematics, Imperial College, London, United Kingdom

Received 15 November 1991
Revised 31 January 1992

Abstract

Cash, J.R. and H.H.M. Silva, On the numerical solution of a class of singular two-point boundary value problems, Journal of Computational and Applied Mathematics 45 (1993) 91–102.

Mono-implicit Runge–Kutta (MIRK) formulae used in a deferred correction framework have proved to be a very effective way of solving first-order systems of nonlinear two-point boundary value problems. We examine this general approach in detail and question whether MIRK formulae are the most efficient class of Runge–Kutta methods to use. We also consider the extension of this deferred correction appproach to deal with problems having a certain type of singularity at the end of the range of integration. We consider the questions whether it is possible to derive MIRK formulae which are applicable to such problems and whether it is, in general, possible to solve such problems automatically. We will answer the first question in the affirmative. However we will show that, despite claims to the contrary appearing in the literature, the answer to the second question is no.

Keywords: Mono-implicit Runge–Kutta formulae; two-point boundary value problems; deferred correction; singularities of the first kind.

1. Introduction

In [9,10] a new class of implicit Runge–Kutta formulae, known as mono-implicit (MIRK) formulae, was proposed for the numerical solution of first-order systems of ordinary differential

Correspondence to: Prof. J.R. Cash, Department of Mathematics, Imperial College, South Kensington, London SW7 2AZ, United Kingdom.

* This author's work is supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico — CNPq of the Ministry for Science and Technology of Brazil under grant 201208/89.0.

equations. In particular, in [10], MIRK formulae of order 4, 6, 8 were derived for the numerical solution of the general first-order, two-point boundary value problem

$$\frac{dy}{dx} = f(x, y), \quad a \leqslant x \leqslant b, \qquad g(y(a), y(b)) = 0, \quad y \in \mathbb{R}^s. \tag{1.1}$$

These formulae were specially designed to have the very good stability necessary to deal with challenging boundary value problems, in particular those with sharp boundary layers or turning points. Although MIRK formulae have some important computational advantages, they suffer from the usual drawback associated with Runge–Kutta formulae. This is that as the order of the formula increases, the computational effort required to compute the solution also increases considerably. The effectiveness of MIRK formulae for solving two-point boundary value problems was improved considerably in [7,8,11] where a deferred correction framework based on these formulae was developed. It has been apparent for some time that the use of iterated deferred corrections allows high-order formulae to be implemented at a moderate computational cost. However, an important advance described in [8] is to allow the possibility of having an implicit correction term. In this way we can avoid the loss of stability normally associated with deferred correction. Since we are mainly interested in singular perturbation problems (indeed in problems in general having nonsmooth parts to their solution trajectories), it is very important that stability be preserved when carrying out deferred correction.

The main theory behind our particular deferred correction approach is based on a result of [12] and in what follows we describe the way in which the result is used. To explain this we suppose that we wish to generate the numerical solution of (1.1) on the mesh

$$\Pi: \quad a = x_0 < x_1 < \cdots < x_N = b.$$

We denote the restriction of the continuous solution $y(x)$ to the grid Π by the vector Δy. For ease of presentation we assume that the mesh is uniform so that $x_{i+1} - x_i = h$, $0 \leqslant i \leqslant N-1$. Suppose now that we have available a "cheap" low-order method

$$\phi(\eta) = 0, \tag{1.2}$$

the solution of which gives a first approximation η to Δy. In order for Skeel's theorem [12] to hold, it is necessary to assume that the numerical method ϕ is stable and that the low-order numerical solution η has certain smoothness properties that are given in detail in [9]. We now set up a deferred correction step defined as

$$\phi(\bar{\eta}) = \psi(\eta), \tag{1.3}$$

where $\psi(\eta)$ is an approximation to the local truncation error of ϕ. We denote the solution of (1.3) by $\bar{\eta}$ with the hope of course being that $\bar{\eta}$ is a better approximation than η to Δy. The following theorem gives an order result on $\bar{\eta}$.

Theorem 1.1 (Skeel [12]). *If for arbitrary functions Δw, having at least r continuous derivatives, the following conditions hold:*

(i) $\eta = \Delta y + O(h^p)$,

(ii) $\psi(\Delta y) = \phi(\Delta y) + O(h^{r+p})$,

(iii) $\psi(\Delta w) = O(h^r)$,

it follows that $\bar{\eta} = \Delta y + O(h^{r+p})$.

In [8] it was shown that if ϕ_p, ϕ_{p+2} are symmetric Runge–Kutta formulae of order p, $p+2$, respectively, and if $\psi \equiv \phi_p - \phi_{p+2}$, then $\bar{\eta}$ is normally an $O(h^{p+2})$ approximation to Δy. Indeed conditions (i), (ii) of Skeel's theorem are trivial to satisfy, it is condition (iii) which causes the difficulty. Even if we restrict the deferred correction term ψ to having this particular form, there is still quite a wide choice of candidates for ϕ_p. We have found that mono-implicit Runge–Kutta formulae are particularly good for use in a deferred correction framework. This is because they can be "opened out" and expressed in terms of y_n and y_{n+1} only. This in turn makes the correction term relatively cheap to evaluate once it is given explicitly in terms of these unknowns. Extensive numerical testing [11] has shown that a code, HAGRON, based on MIRK formulae, is very efficient for the numerical solution of a large class of difficult nonlinear two-point boundary value problems. In particular, numerical results presented in [11] show that on a large test set of difficult problems the code HAGRON compares very favourably with the widely used code COLSYS [4] and its newer version COLNEW [1]. It would be wrong to regard these codes as being in competition. Instead we regard HAGRON and COLNEW as complementing each other as effective methods for solving general nonlinear two-point boundary value problems. Even though HAGRON provides a reasonably effective approach to the numerical solution of the first-order system (1.1), it is natural to investigate

(i) ways in which the underlying algorithm can be improved, e.g., by changing the basic integration formulae;

(ii) classes of boundary value problems for which the basic approach is inappropriate with a view to widening the applicability of our code.

In what follows we will consider (i) and (ii).

As explained previously, Skeel's theorem is a very general one. In the code HAGRON we have implemented the particular case where ϕ_p, ϕ_{p+2} are Runge–Kutta methods of order p, $p+2$, respectively and we seek to raise the order of the solution from p to $p+2$ by the application of a single deferred correction. HAGRON implements MIRK formulae although Skeel's theorem will carry through providing that ϕ_p, ϕ_{p+2} are any consistent symmetric Runge–Kutta formulae. It is therefore an obvious extension to consider the possibility of ϕ_p and ϕ_{p+2} being fully implicit Runge–Kutta formulae other than MIRK formulae. Using experience from initial-value problems, two natural classes of formulae to consider are singly implicit formulae and symmetric Runge–Kutta formulae based on Gauss points. As with initial-value problems, there is a trade-off between different classes of formulae as regards computational effort, accuracy, storage requirements, etc. This trade-off is well known and widely understood for IVPs but has received very little attention in the case of BVPs. It is also the case that methods for BVPs are likely to benefit much more by the use of parallel processors than are methods for IVPs. It would therefore be very valuable to compare a deferred correction scheme based on Gauss point Runge–Kutta formulae with one based on MIRK formulae and we are in the process of carrying out such an investigation. In addition there is the possibility of mixing formulae in the various stages of the deferred correction algorithm, for example ϕ_p could be a MIRK formula with ϕ_{p+2} being a Gauss formula. This approach is also under investigation.

There is an important class of boundary value problems that COLSYS claims to deal with effectively but which cannot at present be handled automatically by HAGRON. These are problems posed in second- (or higher-) order form which have a certain type of singularity, normally occurring at one end of the range of integration. Such problems are typified by the

particle diffusion and reaction equation given in [5, p.8]:

$$\begin{aligned} y'' + \frac{2}{x}y' &= -\phi^2 \beta C \, e^{\gamma(1-y^{-1})}, \\ C'' + \frac{2}{x}C' &= \phi^2 C \, e^{\gamma(1-y^{-1})}, \end{aligned} \tag{1.4}$$

$0 \leqslant x \leqslant 1$, with boundary conditions at $x = 0$ being $y'(0) = C'(0) = 0$. Note that in the formulation of (1.4) we have a term y'/x which has the indeterminate value $0/0$ at $x = 0$. It is clear that a code used to solve (1.4) should not attempt to evaluate y'' at $x = 0$. This precludes the use of a large class of Runge–Kutta formulae, namely formulae using Lobatto or Radau points and in particular MIRK formulae. In addition, shooting methods will experience difficulties if they attempt to evaluate the derivative term at the beginning of the range of integration. The term y'/x can often be handled analytically, by means of a series expansion for example, but for general problems this can be inconvenient (although for some problems it may be unavoidable). Another possibility would be to change the formula being used when we are "close" to the singularity, but this is also an inconvenient approach. Ideally we would like to be able to deal with such problems automatically without the need for user intervention. There have been several claims [3–5] that the code COLSYS is able to do this because it is based on Gauss point Runge–Kutta formulae which do not evaluate the derivative terms at the end of the integration range. The start of our investigations was to derive a class of modified MIRK formulae which did not call for the evaluation of derivatives at the end of the integration range and which would fit naturally into a deferred correction framework.

2. Modified MIRK formulae

In this section we give the motivation for, and the definition of, the concept of a modified mono-implicit Runge–Kutta formula. Such formulae can be expressed solely in terms of y_n and y_{n+1} but differ from MIRK formulae in that they are not based on Lobatto points. In [10], MIRK formulae were regarded as modified linear multistep methods and their accuracy was investigated by means of Taylor expansions. However, this approach is inconvenient for modified formulae. In view of this we will cast our formulae into a Runge–Kutta framework and use the extensive theory available. Since we are interested only in the numerical solution of two-point boundary value problems, we will confine our attention to symmetric formulae. It has been shown [5] that such formulae are the best candidates for having the excellent stability necessary for the efficient solution of stiff boundary value problems.

2.1. A two-stage, second-order formula

This has the general form

$$y_{n+1} - y_n = h\left[b_1 f(x_{n+\alpha}, \bar{y}_{n+\alpha}) + b_1 f(x_{n+1-\alpha}, \bar{y}_{n+1-\alpha})\right],$$

where

$$\bar{y}_{n+\alpha} = A y_{n+1} + (1-A) y_n, \qquad \bar{y}_{n+1-\alpha} = (1-A) y_{n+1} + A y_n.$$

Putting this into a standard Runge–Kutta framework and imposing the condition that the method should have order 2, we get

$$\begin{array}{c|cc} \alpha & \frac{1}{2}\alpha & \frac{1}{2}\alpha \\ 1-\alpha & \frac{1}{2}(1-\alpha) & \frac{1}{2}(1-\alpha) \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{2.1}$$

Note that we can save one function evaluation by taking $\alpha = \frac{1}{2}$. In this case our formula reduces to the implicit mid-point rule. Another special case is $\alpha = 0$, which yields the trapezium rule.

2.2. A four-stage, fourth-order formula

This has the form

$$\begin{aligned} y_{n+1} - y_n = h\big[& b_1\big(f(x_{n+\alpha}, \bar{y}_{n+\alpha}) + f(x_{n+1-\alpha}, \bar{y}_{n+1-\alpha})\big) \\ & + b_3\big(f(x_{n+\beta}, \bar{y}_{n+\beta}) + f(x_{n+1-\beta}, \bar{y}_{n+1-\beta})\big)\big], \end{aligned}$$

where

$$\begin{aligned} &\bar{y}_{n+\alpha} = \alpha y_{n+1} + (1-\alpha) y_n, \qquad \bar{y}_{n+1-\alpha} = \alpha y_n + (1-\alpha) y_{n+1}, \\ &\bar{y}_{n+\beta} = A y_{n+1} + (1-A) y_n + h\big[Cf(x_{n+\alpha}, \bar{y}_{n+\alpha}) + Df(x_{n+1-\alpha}, \bar{y}_{n+1-\alpha})\big], \\ &\bar{y}_{n+1-\beta} = (1-A) y_{n+1} + A y_n - h\big[Df(x_{n+\alpha}, \bar{y}_{n+\alpha}) + Cf(x_{n+1-\alpha}, \bar{y}_{n+1-\alpha})\big]. \end{aligned}$$

Writing this in standard Runge–Kutta notation, we have

$$\begin{array}{c|cccc} \alpha & \alpha b_1 & \alpha b_1 & \alpha b_3 & \alpha b_3 \\ 1-\alpha & (1-\alpha)b_1 & (1-\alpha)b_1 & (1-\alpha)b_3 & (1-\alpha)b_3 \\ \beta & Ab_1 + C & Ab_1 + D & Ab_3 & Ab_3 \\ 1-\beta & (1-A)b_1 - D & (1-A)b_1 - C & (1-A)b_3 & (1-A)b_3 \\ \hline & b_1 & b_1 & b_3 & b_3 \end{array}$$

Again we would have the possibility of saving one function evaluation by taking $\beta = \frac{1}{2}$. This now suggests the general form that should be taken by a modified symmetric MIRK formula. We will write this formula, using standard Runge–Kutta notation, as

$$\begin{array}{c|c} \boldsymbol{C} & A \\ \hline & \boldsymbol{b} \end{array}$$

where $\boldsymbol{b}, \boldsymbol{C} \in \mathbb{R}^s$ and A is $s \times s$. Assuming that the integer s is even, the vectors b and C have the following form:

$$b_{2i} = b_{2i-1}, \quad C_{2i} = 1 - C_{2i-1}, \qquad 1 \leqslant i \leqslant \tfrac{1}{2}s.$$

The coefficient matrix A has the special form

$$\begin{aligned} &A_{1,i} = C_1 b_i, && A_{2,i} = (1 - C_1) b_i, \\ &A_{2i+1,j} = A_i b_j + B_{2i+1,j}, && A_{2i+2,j} = (1 - A_i) b_j - C_{2i+1,j}, \end{aligned}$$

where $B_{2i+1,j} = C_{2i+1,j} = 0$ for $j \geqslant 2i+1$, and

$$C_{2i+1,j} = \begin{cases} -B_{2i+1,j+1}, & j \text{ odd}, \\ -B_{2i+1,j-1}, & j \text{ even}. \end{cases}$$

Here the b_i, C_i and $B_{i,j}$ are arbitrary coefficients to be chosen to satisfy the various-order relations. In general we can save one function evaluation by choosing $C_{2s-1} = C_{2s} = \frac{1}{2}$. Having given the definition of a modified MIRK formula, we go on to consider some particular formulae in the next section.

3. Some particular formulae

In this section we will derive some modified MIRK formulae of order 4 and 6. The question concerning the minimum number of stages required to give a MIRK formula of order p is an interesting one which is still open. We will derive a three-stage, fourth-order formula and a seven-stage sixth-order one. Certainly there does not exist a fourth-order MIRK formula with less than three stages. There may exist a sixth-order formula with six stages or less, but we have not been able to find one.

3.1. A fourth-order, three-stage formula

A symmetric, three-stage, modified MIRK formula has the form

$$\begin{array}{c|ccc} \alpha & \alpha b_1 & \alpha b_1 & \alpha(1-2b_1) \\ 1-\alpha & (1-\alpha)b_1 & (1-\alpha)b_1 & (1-\alpha)(1-2b_1) \\ \frac{1}{2} & \frac{1}{2}b_1 + t & \frac{1}{2}b_1 - t & \frac{1}{2} - b_1 \\ \hline & b_1 & b_1 & 1-2b_1 \end{array} \tag{3.1}$$

Using standard Runge–Kutta analysis, we can write down the order relations to be satisfied by a fourth-order formula with this structure. Because of the very special structure of this modified MIRK formula, many of the order relations are already satisfied. It can be shown that (3.1) has order 4 if and only if the following two relations hold:

$$b_1\left(\tfrac{1}{2} - 2\alpha + 2\alpha^2\right) = \tfrac{1}{12}, \qquad (1-2b_1)(1-2\alpha)t = \tfrac{1}{12}.$$

A convenient choice is $\alpha = \frac{1}{6}$, which has the advantage of making all the weights positive and gives the final formula

$$\begin{array}{c|ccc} \frac{1}{6} & \frac{1}{16} & \frac{1}{16} & \frac{1}{24} \\ \frac{5}{6} & \frac{5}{16} & \frac{5}{16} & \frac{5}{24} \\ \frac{1}{2} & \frac{11}{16} & -\frac{5}{16} & \frac{1}{8} \\ \hline & \frac{3}{8} & \frac{3}{8} & \frac{1}{4} \end{array} \tag{3.2}$$

3.2. A sixth-order, seven-stage formula

This has the general form

α	αb_1	αb_1	αb_3	αb_3	αb_5	αb_5	αb_7
$1-\alpha$	$(1-\alpha)b_1$	$(1-\alpha)b_1$	$(1-\alpha)b_3$	$(1-\alpha)b_3$	$(1-\alpha)b_5$	$(1-\alpha)b_5$	$(1-\alpha)b_7$
β	Ab_1+S	Ab_1+T	Ab_3	Ab_3	Ab_5	Ab_5	Ab_7
$1-\beta$	$(1-A)b_1-T$	$(1-A)b_1-S$	$(1-A)b_3$	$(1-A)b_3$	$(1-A)b_5$	$(1-A)b_5$	$(1-A)b_7$
γ	Bb_1+Z_1	Bb_1+Z_2	Bb_3+U	Bb_3+V	Bb_5	Bb_5	Bb_7
$1-\gamma$	$(1-B)b_1-Z_2$	$(1-B)b_1-Z_1$	$(1-B)b_3-V$	$(1-B)b_3-U$	$(1-B)b_5$	$(1-B)b_5$	$(1-B)b_7$
$\frac{1}{2}$	$\frac{1}{2}b_1-d_3$	$\frac{1}{2}b_1+d_3$	$\frac{1}{2}b_3-d_2$	$\frac{1}{2}b_3+d_2$	$\frac{1}{2}b_5-d_1$	$\frac{1}{2}b_5+d_1$	$\frac{1}{2}b_7$
	b_1	b_1	b_3	b_3	b_5	b_5	b_7

When solving the order relations for this formula to have order 6, it is convenient to make the standard simplifying assumptions. In particular, we would wish to impose the conditions

$$\Sigma a_{ij}C_j = \tfrac{1}{2}C_i^2, \quad 1 \leqslant i \leqslant 7.$$

Examining this condition for $i = 1$, we find it reduces to the requirement

$$\tfrac{1}{2}\alpha = \tfrac{1}{2}\alpha^2, \quad \text{i.e., } \alpha = 0, 1.$$

It is precisely these values of α which we do not allow for use with modified MIRK formulae, since the choice of $\alpha = 0$ or 1 would call for the derivative to be computed at the ends of the range of integration. The inability to make this implifying assumption causes the order relations to be much harder to solve than would normally be the case. In an attempt to ameliorate this problem we choose $b_1 = b_3 = 0$. Now by solving the order relations, we can recursively define the coefficients of our formula. Again, because MIRK formulae have such a special structure, many of the order relations depend on each other and there are far less than thirty-seven independent relations. (Thirty-seven is the number of order relations that normally need to be solved in order to define a sixth-order formula.) Below we give the coefficients of our sixth-order Runge–Kutta formula. b_1, b_3 are arbitrary. We choose $b_1 = b_3 = 0$, $\gamma = \frac{1}{2} - (\frac{3}{20})^{1/2}$, $b_5 = \frac{5}{18}$, $b_7 = \frac{4}{9}$, α, β arbitrary,

$$T = \frac{\left(\frac{1}{6}\beta^3 - \frac{1}{6}\beta\right)\left(\alpha - \frac{1}{2}\right) - \left(\frac{1}{2}\beta^2 - \frac{1}{2}\beta\right)\left(\frac{1}{2}\alpha^2 - \frac{1}{6}\right)}{\left(\alpha - \frac{1}{2}\right)\left[\frac{1}{2}(1-\alpha)^2 + \frac{1}{2}\alpha^2 - \frac{1}{3}\right]},$$

$$S = \frac{\frac{1}{2}\beta^2 - \frac{1}{2}\beta}{\alpha - \frac{1}{2}} + T, \qquad A = \beta - S - T.$$

Choose z_1, z_2 arbitrarily. We choose $z_1 = z_2 = 0$,

$$V = \frac{\left(\frac{1}{6}\gamma^3 - \frac{1}{6}\gamma\right)\left(\beta - \frac{1}{2}\right) - \left(\frac{1}{2}\gamma^2 - \frac{1}{2}\gamma\right)\left(\frac{1}{2}\beta^2 - \frac{1}{6}\right)}{\left(\beta - \frac{1}{2}\right)\left[\frac{1}{2}(1-\beta)^2 + \frac{1}{2}\beta^2 - \frac{1}{3}\right]},$$

$$U = \frac{\frac{1}{2}\gamma^2 - \frac{1}{2}\gamma}{\beta - \frac{1}{2}} + V, \qquad B = \gamma - U - V.$$

Choose d_3 arbitrarily. We choose $d_3 = 0$. Choose d_1, d_2 to satisfy the two equations (which are linear in d_1 and d_2)

$$\Sigma b_i a_{ij} c_j^3 = \tfrac{1}{20}, \qquad 0 = \tfrac{1}{16} + d_1\left(\tfrac{1}{2} - \gamma\right) + d_2\left(\tfrac{1}{2} - \beta\right).$$

This then satisfies all of the sixth-order relations and gives a seven-stage sixth-order modified MIRK formula.

We could continue this process and derive an eighth-order formula using this Runge–Kutta framework. However, we consider this to be beyond the scope of the present paper. Instead, what we will do in the next section is to consider the stability properties of our formulae and describe how the formulae can be implemented efficiently in a deferred correction setting.

4. Stability

In this section we investigate the absolute stability of our modified MIRK formulae. We do this in the usual way by considering the solution of the scalar test equation $y' = \lambda y$ where λ is a complex constant. Our aim will be to derive a deferred correction formula which has the excellent stability properties necessary for dealing with difficult two-point boundary value problems.

4.1. The fourth-order formula (3.2)

Applying this formula to the test equation $y' = \lambda y$, it is easy to show that

$$\frac{y_{n+1}}{y_n} = \frac{1 + \frac{1}{2}q + \frac{1}{12}q^2}{1 - \frac{1}{2}q + \frac{1}{12}q^2} \equiv R(q), \quad \text{where } q = h\lambda. \tag{4.1}$$

Since $R(q)$ is the (2, 2) Padé approximation to e^q, it follows immediately that formula (3.2) is A-stable. Furthermore, since $R(q)$ is a diagonal Padé approximation, it follows that (3.2) is precisely A-stable.

4.2. The sixth-order formula

Applying the sixth-order formula to $y' = \lambda y$, we find after some straightforward but tedious algebra that

$$\frac{y_{n+1}}{y_n} = \frac{1 + r_1 q + r_2 q^2 + r_3 q^3 + r_4 q^4}{1 - r_1 q + r_2 q^2 - r_3 q^3 + r_4 q^4}, \tag{4.2}$$

where

$$r_1 = \tfrac{1}{2}, \qquad r_2 = b_5(U - V)(1 - 2A) + b_7 d_1(2B - 1) + b_7 d_2(2A - 1),$$
$$r_3 = b_5(UT - VS + US - VT) - b_7 d_1(U + V) - b_7 d_2(T + S),$$
$$r_4 = b_7 d_1(UT - VS - US + VT)(1 - 2\alpha).$$

Using the Runge–Kutta order relations, it is easy to show that these expressions reduce to

$$\frac{y_{n+1}}{y_n} = \frac{1 + \frac{1}{2}q + \frac{11}{60}q^2 + \frac{1}{20}q^3 + \frac{1}{144}q^4}{1 - \frac{1}{2}q + \frac{11}{60}q^2 - \frac{1}{20}q^3 + \frac{1}{144}q^4}, \tag{4.3}$$

and it is straightforward to show that this formula is precisely A-stable. It is also easy to verify that the right-hand side of this expression is an $O(h^7)$ approximation to e^q.

Both of these two MIRK formulae have the excellent stability necessary to deal effectively with the numerical solution of challenging two-point boundary value problems. However, the sixth-order formula is expensive to implement, since it is heavily implicit in the unknowns y_n and y_{n+1}. In view of this we will use our two formulae in a deferred correction framework. The stability of deferred correction schemes was analysed in detail in [8]. However, we will not go into any more detail here, since, as we will see in the next section, our codes are much less successful than we would have hoped on the problems under consideration.

5. Numerical results

By defining the concept of a modified MIRK formula, we have been able to derive a deferred correction method which does not call for the evaluation of derivative values at the end of a step $[x_n, x_{n+1}]$. We quickly found that the performance of this algorithm was erratic. However, since there is a singular term in the differential equation, we might expect certain nonstandard behaviour from the integration formulae. This is indeed the case in practice and we were relieved to find that this erratic behaviour also occurred with COLSYS! Initially we found problems for which COLSYS experienced an order reduction. This seems to be the most common case and we attempted to modify COLSYS to take this into account. However, we also found problems for which there seemed to be no increase in accuracy as the mesh was refined. Also we found problems for which COLSYS accepts totally incorrect solutions. Thus we were led to the important conclusion that this class of problems cannot be handled automatically in general. For linear problems it may be the case that, even if the required solution is smooth, unwanted complementary functions are sufficiently poorly behaved so as to wreck the numerical solution. However, given a particular problem of this singular type, we cannot expect to say beforehand whether a code will be successful. What we need is

(i) a scheme for automatically dealing with order reduction when it occurs;

(ii) an a posteriori test for determining whether an accepted "solution" is indeed a solution to the given problem.

Problem (i) should be straightforward to deal with. There are two obvious possibilities for dealing with (ii). The first comes from the fact that we now have two efficient approaches for dealing with these singular problems. A comparison of the results obtained using these two different methods will sometimes allow us to reject incorrect solutions. However this is far from foolproof. A second approach would be to construct a *continuous* solution from the finally accepted discrete solution (COLSYS already gives a continuous solution of this type). The procedure then would be to see how well this continuous solution satisfies the original differential equation. Research along these lines is at present in progress.

Thus it would seem that for singular problems of the type described in this paper the best we can do is to apply the two methods we have described and see whether convergence to a solution is obtained. If no convergence is obtained, we need an analytic approach to deal with the singularity. If convergence is obtained, we need an a posteriori process to confirm that the "correct solution" has indeed been computed. We conclude this section with some test problems which illustrate the kind of behaviour we have been describing. All of the problems were solved by first converting them to a first-order system. We do not give any results for the deferred correction approach, since it did not solve any problems which COLSYS failed to solve and its general behaviour was similar to that of COLSYS.

Problem 5.1.

$$y'' - \frac{1}{x^2}y' = \cos x - \frac{1}{x^2}\sin x, \quad y'(0) = 0, \quad y(1) = 1 - \cos 1.$$

(Solution $y(x) = 1 - \cos x$.) This solution was obtained very easily even though $y'/x^2 = \sin x/x^2$ is unbounded at $x = 0$.

Problem 5.2.

$$y'' + 100y - \frac{y'}{x} = 1000x + \frac{10\cos 10x - 10}{x}, \quad y'(0) = 0, \quad y(1) = 10 - \sin 10.$$

(Solution $y(x) = 10x - \sin 10x$.) This equation was run with a requested accuracy of 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}. The results obtained are given in Table 1. Here "Tol" is the requested accuracy, "Time" is the CPU-time on an IBM RS in double precision and "Points" is the number of points in the grid chosen by COLSYS. When solving the problem with a tolerance of 10^{-8}, we computed the error in $y(0)$ and the ratio of the corresponding error each time the number of grid intervals was doubled. These results are given in Table 2. Since, for nonsingular problems, COLSYS is an eighth-order method, we might hope for similar behaviour on Problem 5.2.

However, these results suggest very strongly that COLSYS is behaving like a fifth-order method for this problem, having suffered an order reduction due to the singular term. A

Table 1

Tol	Time	Points
10^{-2}	0.183	10
10^{-4}	0.350	20
10^{-6}	1.517	80
10^{-8}	3.333	160

Table 2

Points	Error in $y(0)$	Ratio
5	$0.62\cdot10^{-2}$	
10	$0.24\cdot10^{-3}$	25.78
20	$0.79\cdot10^{-5}$	30.48
40	$0.25\cdot10^{-6}$	31.63
80	$0.79\cdot10^{-8}$	31.93
160	$0.25\cdot10^{-9}$	31.95

Table 3

Mesh points	Estimated error
10	$0.35\cdot10^{-6}$
20	$0.39\cdot10^{-7}$
40	$0.69\cdot10^{-6}$
80	$0.54\cdot10^{-4}$
160	$0.89\cdot10^{0}$

Frobenius expansion shows that this problem has a complementary function behaving like $x^2 \ln x$ for small x and it is this that is causing the difficulty.

The third problem we consider is the following.

Problem 5.3.

$$y'' - \frac{1}{x^2}y' = \sin x + \frac{\cos x - 1}{x^2}, \quad y'(0) = 0, \; y(1) = 1 - \sin 1.$$

(True solution is $y(x) = x - \sin x$.) For a requested accuracy of 10^{-2}, 10^{-4}, 10^{-6}, COLSYS obtains the correct solution. For 10^{-8} the estimated error is as in Table 3. This is a fairly common example where a reduced mesh leads to less accuracy.

Finally, we give an example of problems for which an incorrect solution is accepted. The problems of interest are the following.

Problem 5.4.

$$\epsilon^2 y'' - 4\epsilon^2 \frac{y'}{x} - \left(1 - \frac{6\epsilon^2}{x^2}\right) y = 0, \quad y'(0) = 0, \; y(1) = 1.$$

(True solution $y(x) = x^2 e^{(1-x)/\epsilon}$, $\epsilon = 0.1$.)

Problem 5.5.

$$y'' + 100y - \frac{y'}{x^2} = 1000x - \frac{10}{x^2} + \frac{10 \cos 10x}{x^2}, \quad y'(0) = 0, \; y(1) = 10 - \sin 10.$$

(Solution $y(x) = 10x - \sin 10x$.)

For both of these problems and at all tolerances COLSYS accepted a solution which had no resemblance to the true solution. This highlights the need for an a posteriori test to determine whether the computed "solution" is indeed the true solution when solving singular problems.

Finally, we remark that we should not be surprised by these results, since the singular TPBVPs considered in this section, apart from Problem 5.2, are not covered by any serious mathematical theory. Problem 5.2 is normally referred to as a *singular problem of the first kind* (i.e., the singularity is of the form $1/x$). Here the theory leads us to expect reasonable behaviour from COLSYS except that superconvergence is normally lost at the collocation points [5]. The other problems have a more severe singular behaviour and this probably explains the sometimes poor performance of COLSYS and deferred correction. One of the practical difficulties we need to address for Problems 5.4 and 5.5 is "How well does the COLSYS solution satisfy the problem?". This is not an easy question to answer, but may well be the key to solving these problems numerically.

References

[1] U. Ascher and G. Bader, A new basis implementation for a mixed order boundary value O.D.E. solver, *SIAM J. Sci. Statist. Comput.* **8** (1987) 483–500.
[2] U. Ascher, J. Christiansen and R.D. Russell, COLSYS — A collocation code for boundary value problems, in: B. Childs, M. Scott, J.W. Daniel, E. Derman and P. Nelson, Eds., *Codes for Boundary Value Problems in ODEs*, Lecture Notes in Comput. Sci. **76** (Springer, Berlin, 1979) 164–185.
[3] U. Ascher, J. Christiansen and R.D. Russell, A collocation solver for mixed order systems of boundary value problems, *Math. Comp.* **33** (1979) 659–679.
[4] U. Ascher, J. Christiansen and R.D. Russell, Collocation software for boundary value ordinary differential equations, *ACM Trans. Math. Software* **7** (1981) 209–222.
[5] U. Ascher, R.M.M. Mattheij and R.D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1988).
[6] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations* (Wiley, Chichester, 1987).
[7] J.R. Cash, On the numerical integration of non-linear two-point boundary value problems using iterated deferred corrections, Part 1: A survey and comparison of some one-step formulae, *Comput. Math. Appl.* **12a** (1986) 1029–1048.
[8] J.R. Cash, Part II, The development and analysis of highly stable deferred correction formulae, *SIAM J. Numer. Anal.* **25** (1988) 862–882.
[9] J.R. Cash and A. Singhal, Mono-implicit Runge–Kutta formulae for the numerical integration of stiff differential equations, *IMA J. Numer. Anal.* **2** (1982) 211–227.
[10] J.R. Cash and A. Singhal, High order methods for the numerical solution of two-point boundary value problems, *BIT* **22** (1982) 184–199.
[11] J.R. Cash and M.H. Wright, A deferred correction method for nonlinear two-point boundary value problems: Implementation and numerical evaluation, *SIAM J. Sci. Statist. Comput.* **12** (1991) 971–989.
[12] R.D. Skeel, A theoretical foundation for proving accuracy results for deferred corrections, *SIAM J. Numer. Anal.* **19** (1982) 171–196.